ภาษาเครื่องสำหรับภาษาสูง

ตอน 2 *โปรแกรมภาษาเครื่องสำหรับภาษาเครื่อง!*

บุญเลิศ เอี่ยมทัศนา

ในตอนที่ 1 ได้กล่าวถึงเรื่องราวของซีพียูโดยสังเขป ในตอนนี้จะนำความรู้จากตอนที่แล้วมาเขียนโปรแกรม ซึ่งการเขียนโปรแกรมในตอนนี้จะหนักทางด้านภาษาเครื่องสำหรับภาษาเครื่องก่อน เพื่อให้รับทราบเรื่องราวบางเรื่องเป็นพื้นฐานสำหรับเขียนโปรแกรมภาษาเครื่องสำหรับภาษาสูงต่อไป

โครงงานตัวอย่าง

เพื่อแสดงถึงวิธีการในการเขียนโปรแกรมภาษาเครื่อง ผมเห็นว่าควรจะยกโครงงานขึ้นมาเป็นตัวอย่างสักโครงงานหนึ่ง จะทำให้เข้าใจดีกว่ากล่าวถึงวิธีการโดยตรง แต่พอนึกถึงโครงงานที่จะนำมาเป็นตัวอย่างก็อึดอัดใจ เพราะผมต้องการโครงงานที่อยู่ใกล้ตัวทุกคนรู้จักอยู่แล้วและเขียนโปรแกรมได้ง่ายและสั้นด้วย ผลที่สุดก็หนีไม่พ้นเรื่องคีย์บอร์ดและจอภาพ

เมื่อคุณกดคีย์จะปรากฏภาพอักขระของคีย์ที่กดบนจอภาพ คุณจะสงสัยหรือไม่ก็ตามผมขอเรียนให้ทราบว่า โปรแกรมที่ทำงานนี้จะแบ่งเป็น 2 ส่วน ส่วนหนึ่งจะอ่านค่าจากคีย์บอร์ด อีกส่วนหนึ่งจะเป็นการนำไปแสดงบนจอภาพ ซึ่งส่วนมากจะทำเป็นโปรแกรมย่อย 2 ตัวเพื่อใช้ในงานอื่นด้วย สมมติว่าให้ชื่อว่า getchar และ dspchar ตามลำดับ โดยให้

getchar ตรวจจับการกดคีย์ (หมายถึงว่าหากเรียกใช้โปรแกรมนี้จะตรวจจับการกดคีย์จนกว่าผู้ใช้จะกดคีย์) ได้ค่าคีย์ที่กดเป็นรหัส ASCII ในแอคคิวฯ A ส่วน dspchar จะนำค่าในแอคคิวฯ A ออกแสดงบนจอภาพและเลื่อนเคอร์เซอร์ แต่เนื่องจาก getchar จะเกี่ยวข้องกับฮาร์ดแวร์มากพอสมควรจึงของดกล่าว โครงงานนี้จึงจะเพียงแต่แสดงถึงการเขียนโปรแกรม dspchar (Display character) เท่านั้น

ศึกษาระบบ

การเขียนโปรแกรมภาษาเครื่องก็เช่นเดียวกับการเขียนโปรแกรมในภาษาสูง คือมีขั้นตอนซึ่งแบ่งกันไว้หลายแบบ สำหรับผมแบ่งออกเป็น ศึกษาหรือวิเคราะห์ระบบ วางแนวความคิด เขียนร่างโปรแกรม เขียนโปรแกรม นำไปติดตั้ง(คำนี้กว้างขวางมาก ในที่นี้หมายถึงการป้อน ตลอดเลยไปถึงให้รันโปรแกรมนี้ในเครื่องที่ต้องการได้) การค้นหาที่ผิดและแก้(debug) และการปรับปรุงเปลี่ยนแปลง แต่ในบทความตอนนี้ผมจะกล่าวจนถึงขั้นการป้อนโปรแกรมเท่านั้น

จากโครงงานตัวอย่าง ในเมื่อซีพียูจำลอง เครื่องคอมพิวเตอร์ของผมก็เป็น

คอมพิวเตอร์จำลองด้วย โดยให้ออกจอภาพในแบบ 80 อักขระ × 25 บรรทัด โดยที่ผมจัดหน่วยความจำตั้งแต่แอดเดรส $C000 ถึง $C7CF เป็นหน่วยความจำสำหรับจอภาพ (video RAM) ซึ่งคุณไม่ต้องสนใจว่าทางด้านฮาร์ดแวร์ทำได้อย่างไร

ขอให้ทราบแต่เพียงว่าถ้าเราให้ค่ารหัส ASCII ของอักษรใดแก่หน่วยความจำที่แอดเดรส $C000 ก็จะเกิดภาพอักขระนั้นที่มุมบนซ้ายของจอภาพ ถ้าให้ที่แอดเดรส $C7CF ก็จะเกิดภาพที่มุมล่างขวา และเพราะในแต่ละบรรทัดมี 80 อักขระคือ $50 ดังนั้นถ้าให้ค่ารหัสที่แอดเดรส $C04F ก็จะเกิดภาพที่มุมบนขวาหรือคือตัวสุดท้ายของบรรทัดแรก ถ้าให้ที่ $C050 ก็จะเกิดภาพเป็นตัวแรกของบรรทัดที่สองและต่อๆ ไป

หากสงสัยว่าตัวเลข $C7CF ได้มาอย่างไร ผมได้มาจาก 80 × 25 = 2000 = $7D0 คือต้องการหน่วยความจำสำหรับจอภาพเป็น $7D0 ไบท์ แต่นับไบท์แรกเป็นศูนย์ หน่วยความจำสำหรับจอภาพจึงเริ่มตั้งแต่ $C000 ถึง $C7CF

วางแนวความคิด

ในการติดตามแนวความคิดของผมต่อไป ขอให้นึกภาพเมื่อผู้ใช้กดคีย์อักษรต่อเนื่องกันไปเรื่อย เริ่มจากมุมบนซ้ายคือตัวแรกของบรรทัดแรก ไม่เกี่ยวกับคีย์ควบคุมอื่น ๆ เช่น RET หรือ ENTER ซึ่งให้ขึ้นบรรทัดใหม่ ถ้าผู้ใช้พิมพ์เกิน 80 อักขระ บนจอภาพจะเห็นเป็นการขึ้นบรรทัดใหม่เช่นกัน แต่ขอเรียกว่าเป็นการตกบรรทัด ตรงนี้ขอทำความเข้าใจด้วยครับว่าบนจอภาพเราเห็นเป็นตกบรรทัด แต่ในหน่วยความจำสำหรับจอภาพคงเก็บรหัสเรียงติดต่อกันไป เช่นที่แอดเดรส $C04F กับ $C050 ในหน่วยความจำมันต่อกันอยู่ แต่บนจอภาพเห็นเป็นตกบรรทัด

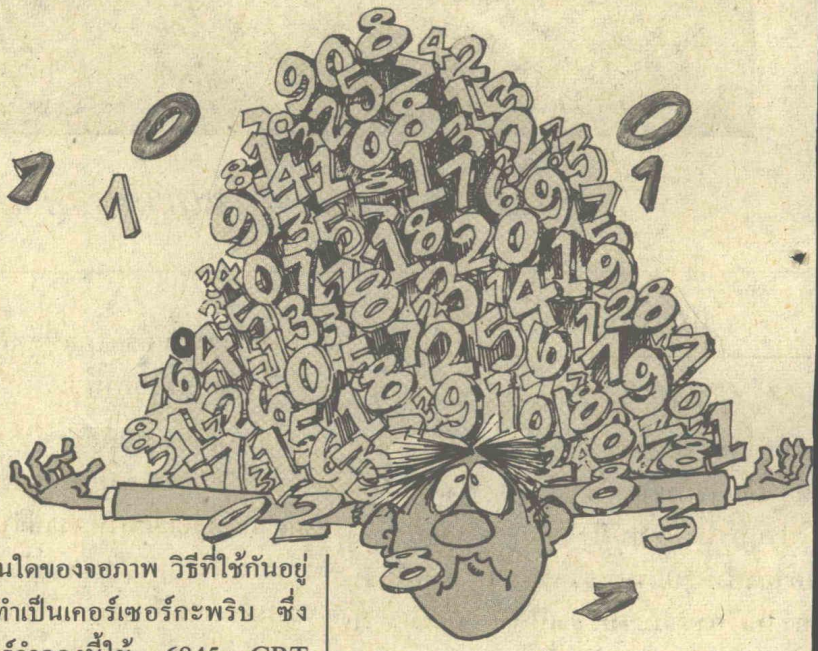
โดยที่โปรแกรมนี้เป็นโปรแกรมที่จะถูกเรียกใช้ผ่านไป ซึ่งในการเรียกใช้แต่ละครั้งจะมีการเปลี่ยนแปลงค่าบางค่า และเราต้องเก็บค่านี้ไว้สำหรับการเรียกใช้ในครั้งต่อไป ซึ่งจะเก็บค้างไว้ในรีจิสเตอร์ไม่ได้และไม่สะดวกที่จะเก็บลงสแต็ค จึงเหลือวิธีเดียวคือเก็บไว้ในหน่วยความจำโดยตรง

ผมจะจัดหน่วยความจำหน่วยหนึ่งขนาด 2 ไบท์ ให้ชื่อว่า SCRPNTR อยู่ที่แอดเดรสใดยังไม่กำหนด แต่จะให้ค่าเริ่มต้นเป็น $C000 เมื่อ DSPCHAR ถูกเรียกใช้จะนำค่าในแอคคิวฯ A ไปเก็บในหน่วยความจำที่ SCRPNTR ชี้อยู่ (Indirect Addressing Mode) คือถ้าเป็นการเรียกใช้ครั้งแรกจะเก็บที่ $C000 แล้วเพิ่มค่า SCRPNTR ขึ้นหนึ่งเป็น $C001 นั่นคือถ้าเรียกใช้ครั้งที่ 2 จะเก็บที่ $C001 และต่อ ๆ ไป ซึ่งจะปรากฏภาพที่บรรทัดแรกตัวที่ 1, ที่ 2 และต่อ ๆ ไปเช่นกัน

แต่การที่จะแสดงภาพอักขระบนจอภาพเราต้องแสดงให้ผู้ใช้ทราบว่าจะปรากฏภาพอักขระตรงส่วนใดของจอภาพ วิธีที่ใช้กันอยู่ในปัจจุบันคือทำเป็นเคอร์เซอร์กะพริบ ซึ่งในคอมพิวเตอร์จำลองนี้ใช้ 6845 CRT Controller ควบคุมจอภาพ จึงทำได้โดยให้ค่า SCRPNTR แก่ไอซีเบอร์นี้ จะทำให้เกิดภาพเคอร์เซอร์ตรงตำแหน่งที่ต้องการ

จากแนวความคิดดังกล่าวนี้เพียงพอแล้วที่จะนำไปเขียนโปรแกรม DSPCHAR แต่ส่วนมากมักจะมีงานฝากจากโปรแกรมอื่น เช่น โปรแกรมที่คอบสนองการกดคีย์ ENTER อาจต้องการค่า column และ row ของเคอร์เซอร์ซึ่งผมจะจัดการให้โดยเก็บไว้ในหน่วยความจำขนาด 1 ไบท์ 2 หน่วย ให้ชื่อว่า COL และ ROW ตามลำดับ โดยให้ค่าเริ่มต้นเป็น $00

มีอีกเรื่องหนึ่งที่ผมไม่ได้กล่าวแต่ต้นคือ การเขียนโปรแกรมภาษาเครื่องในปัจจุบันนิยมเขียนเป็นโมดูลย่อย หลาย ๆ โมดูลรวมกันเป็นโปรแกรม โมดูลหนึ่งๆ อาจเป็นส่วนของโปรแกรมหรือโปรแกรมย่อยก็ได้ และส่วนมากจะมีโมดูลหนึ่งนิยมเรียกกันว่า INIT ทำหน้าที่เตรียมการต่างๆ เช่น ในส่วนที่เกี่ยวกับ DSPCHAR จะมีการเตรียมค่าเริ่มต้นดังกล่าวแล้ว และการลบจอภาพคือให้ค่า $20 แก่หน่วยความจำสำหรับจอภาพ (แอดเดรส $C000 ถึง $C7CF) ในการเขียนโปรแกรมจึงต้องจดไว้ว่าโมดูลใดต้องการให้เตรียมอะไรบ้าง แล้วนำไปใส่ไว้ในโมดูล INIT



ร่างโปรแกรม

การเขียนร่างโปรแกรมนิยมเขียนเป็นคำพูดธรรมดา บ่งว่าต้องการให้ซีพียูทำอะไรโดยไม่คำนึงว่าจะทำได้อย่างไร (แต่มีเรื่องที่จะต้องคำนึงไว้เสมอว่าซีพียูจะทำตามที่เราสั่งไม่ใช่ที่เราคิด) ในการเขียนร่างนี้จะมีการใช้คำสั่งเทียม ซึ่งเราจะต้องนำคำสั่งเทียมไปเขียนขยายความ (คือเป็นการเขียนร่างครั้งที่ 2, 3, 4 และต่อ ๆ

```
โมดูล DSPCHAR
BEGIN
      นำค่าใน A ไปให้แก่ (SCRPNTR)
      เพิ่มหนึ่ง SCRPNTR
      เพิ่มหนึ่ง COL
      IF COL มีค่ามากกว่า 79 THEN
         COL มีค่าเป็นศูนย์
         เพิ่มหนึ่ง ROW
         IF ROW มีค่ามากกว่า 25 THEN
            ให้ ROW มีค่าเป็น 25
            เลื่อนจอภาพขึ้น 1 บรรทัด {ScreenUp}
            ลบค่า SCRPNTR ด้วย 80
         ENDIF
      ENDIF
      เลื่อนเคอร์เซอร์ (MoveCursor)
END;
```

รูปที่ 1 แสดงร่างโปรแกรม DSPCHAR ซึ่งค่าของ SCRPNTR ได้ให้ค่าเริ่มต้นไว้เป็น $C000 ส่วน COL และ ROW ได้ให้ค่าเริ่มต้นไว้เป็น 0 ทั้งคู่

```
Module DSPCHAR
BEGIN
  (SCRPNTR) := ACC;
  SCRPNTR := SCRPNTR + 1;
  COL := COL + 1;
  IF COL > 79 THEN
    COL := 0;
    ROW := ROW + 1;
    IF ROW > 24 THEN
      ROW := 24;
      SCREEN_UP;
      SCRPNTR := SCRPNTR - 80;
    ENDIF
  ENDIF
  MOVE_CURSOR;
END
```

รูปที่ 2 แสดงร่างโปรแกรม DSPCHAR ที่ใช้คำสั่งเทียมตลอด และเป็นร่างโปรแกรมแบบที่จะใช้ในบทความนี้ต่อไป เครื่องหมาย := หมายถึง ให้นำค่าทางขวา (ซึ่งถ้ามีการคำนวณให้นำผลจากการคำนวณ) ไปให้แก่ทางซ้าย

ไป) เข้าหาภาษาที่เราจะใช้ จนเห็นว่าจะเขียนในภาษานั้นได้แล้วซึ่งในกรณีนี้คือภาษาเครื่อง ให้นำไปเขียนโปรแกรม แต่บังเอิญตัวอย่างที่ยกมานี้ไม่จำเป็นต้องเขียนขยายความทั้งในตัวโปรแกรมเอง และผมจะของดเขียนโปรแกรมย่อยที่เป็นโมดูลด้วย จึงเขียนร่างโปรแกรมเพียงครั้งเดียว ดังแสดงในรูปที่ 1

ถึงแม้จะเป็นร่างโปรแกรมที่ใช้คำพูดธรรมดา ก็คงต้องมีคำอธิบายบ้างที่ว่านำค่าใน A ไปให้แก่ (SCRPNTR) หมายถึงหน่วยความจำที่ SCRPNTR ชี้อยู่ (indirect addressing mode) ส่วนการเพิ่มหนึ่ง SCRPNTR หมายถึงการเพิ่มหนึ่งให้แก่ค่าในหน่วยความจำที่กำหนดเรียกว่า SCRPNTR (direct addressing mode) ข้อความในวงเล็บปีกกาหมายถึงหมายเหตุ และที่เขียนไว้คือคำสั่งเทียมเพื่อหมายเหตุไว้ว่าจะนำไปเขียนเป็นโปรแกรมย่อย เรื่องนี้จะเห็นถนัดเมื่อเขียนด้วยคำสั่งเทียมตลอดดังแสดงในรูปที่ 2

หากคุณสงสัยว่าจะทราบได้อย่างไรว่าคำสั่งเทียมใดนำไปเขียนเป็นภาษาเครื่องได้เลย คำสั่งเทียมใดที่ต้องนำไปเขียนเป็นโปรแกรมย่อย ให้ดูจากร่างโปรแกรมจะเห็นว่าส่วนใดที่เป็นการปฏิบัติแสดงว่าสามารถเขียนด้วยภาษาเครื่องได้โดยตรง(ส่วนมาก) ส่วนที่เป็นชื่อมักจะเป็นโปรแกรมย่อย

คำว่า Screen_Up หมายถึงว่า เมื่อผู้ใช้คีย์มาจนถึงปลายบรรทัดที่ 25 คือสุดหน่วยความจำสำหรับจอภาพพอดี ซึ่งบนจอภาพเคอร์เซอร์จะต้องขึ้นบรรทัดใหม่ แต่ไม่มีบรรทัดจะขึ้นจึงต้องเลื่อนภาพทั้งหมดขึ้นไป 1 บรรทัด การเลื่อนบรรทัดนี้หากกล่าวอย่างคร่าว ๆ จะทำได้โดยนำบรรทัดที่ 1 ไปไว้บรรทัดที่ 0 นำบรรทัดที่ 2 ไปไว้ที่บรรทัดที่ 1 และต่อ ๆ ไปจนนำบรรทัดที่ 24 ไปไว้ที่บรรทัดที่ 23 แล้วลบบรรทัดที่ 24 คือให้ค่า $20 แก่หน่วยความจำตั้งแต่แอดเดรส $C780 ถึง $C7CF ซึ่งจากนี้ต่อไปหากผู้ใช้กดคีย์จนตกบรรทัดจะเป็นการเลื่อนบรรทัดทุกครั้ง

เขียนโปรแกรม

ในการเขียนโปรแกรมภาษาเครื่องเพื่อความสะดวกควรจัดหาแบบฟอร์มดังแสดงในรูปที่ 3 (ส่วนมากทำกันเอง เช่น โรเนียวเตรียมไว้) จะเห็นว่าแบ่งเป็น 5 ช่อง สำหรับช่องหมายเหตุ (Comment) มีไว้เพื่อเขียนคำอธิบายเพื่อให้ผู้อ่านโปรแกรมทราบเจตนาของผู้เขียน ซึ่งผมได้นำคำสั่งเทียมจากร่างโปรแกรมมาแสดงไว้ เพราะเห็นว่าชัดเจนและเพื่อให้เทียบว่าเราแปลคำสั่งเทียม

เป็นภาษาเครื่อง แทบจะเป็นบรรทัดต่อบรรทัด

ขั้นตอนในการเขียนได้แสดงในรูปที่ 3, 4 และ 5 ตามลำดับ เริ่มด้วยการนำคำสั่งเทียมในร่างโปรแกรมมาเขียนเป็นนิวมอนิก เสร็จแล้วนำนิวมอนิกไปเปิดตาราง ได้รหัสคำสั่งภาษาเครื่องนำมาใส่ในช่อง Machine code เสร็จแล้วให้แอดเดรส ผมให้แอดเดรสเริ่มต้นเป็น $0740 โดยสมมติว่ามีโปรแกรมอื่นเริ่มต้นมาตั้งแต่แอดเดรส $0000 มาถึง DSPCHAR ที่แอดเดรส $0740 พอดี การให้แอดเดรสส่วนมากจะเขียนด้วยเลข 2 ตัวไปจนจบโปรแกรมย่อยหนึ่ง ๆ กรรมวิธีที่กล่าวมานี้ทั้งหมดเรียกว่าการแอสเซมเบลอร์ด้วยมือ (Hand assembler)

ตอนล่างของโปรแกรมได้บันทึกค่า

ADDR	Machine Code	Label	Mnemonics	Comments
				BEGIN
		DSPCHAR	MOV (SCRPNTR),A	(scrpntr) := acc
			INC SCRPNTR	scrpntr := scrpntr + 1
			INC COL	col := col + 1
			CMP COL,#MAXCOL	IF col > 79 THEN
			BCS DSP1	
			MOV COL,#$00	col := 0
			INC ROW	row := row + 1
			CMP ROW,#MAXROW	IF row > 24 THEN
			BCS DSP1	
			MOV ROW,#MAXROW	row := 24
			CALL SCRUP	ScreenUp
			MOV HL,SCRPNTR	scrpntr := scrpntr - 80
			SUB HL,#$0050	ENDIF
			MOV SCRPNTR,HL	ENDIF
		DSP1	MOV HL,(SCRPNTR)	
			PUSH HL	
			CALL MOVCUR	MoveCursor
		DSP2	RET	END

รูปที่ 3 แสดงการเขียนโปรแกรมภาษาเครื่องลำดับแรก ทำโดยนำคำสั่งเทียมในร่างโปรแกรมมาเขียนเป็นนิวมอนิก (ได้ให้หมายเหตุไว้เป็นคำสั่งเทียมด้วย)

ADDR	Machine Code	Label	Mnemonics	Comments
				BEGIN
	EE AC 80 10	DSPCHAR	MOV (SCRPNTR),A	(scrpntr) := acc
	AF 80 10		INC SCRPNTR	scrpntr := scrpntr + 1
	2F 82 10		INC COL	col := col + 1
	3F 4F		CMP COL,#MAXCOL	IF col > 79 THEN
	75 17		BCS DSP1	
	3E 00		MOV COL,#$00	col := 0
	2F 83 10		INC ROW	row := row + 1
	3F 16		CMP ROW,#MAXROW	IF row > 24 THEN
	75 0E		BCS DSP1	
	3E 16		MOV ROW,#MAXROW	row := 24
	CD __ __		CALL SCRUP	ScreenUp
	3C 80 10		MOV HL,SCRPNTR	scrpntr := scrpntr - 80
	1F 50 00		SUB HL,#$0050	ENDIF
	BC 80 10		MOV SCRPNTR,HL	ENDIF
	EE 3C 80 10	DSP1	MOV HL,(SCRPNTR)	
	F3		PUSH HL	
	CD __ __		CALL MOVCUR	MoveCursor
	F0	DSP2	RET	END

รูปที่ 4 การเขียนโปรแกรมภาษาเครื่องลำดับที่สอง นำนิวมอนิกไปเปิดตารางได้ค่ารหัสคำสั่ง ใส่ในช่อง Machine Code ขอให้สังเกตว่าค่าคงตัวคือ MAXCOL และ MAXROW (ค่าอยู่ในรูปที่ 5) นำมาใส่ตรง ๆ ส่วนตัวแปรใส่ค่าแอดเดรสต้นของมันที่เว้นว่างไว้คือแอดเดรสของโปรแกรมย่อยซึ่งยังไม่ทราบ

```
ADDR | Machine Code |  Label    Mnemonics          |   Comments
0740 | EE AC 80 10  | DSPCHAR   MOV   (SCRPNTR),A  | BEGIN
  44 | AF 80 10     |           INC   SCRPNTR      |  (scrpntr) := acc
  47 | 2F 82 10     |           INC   COL          |  scrpntr := scrpntr + 1
  4A | 3F 4F        |           CMP   COL,#MAXCOL  |  col := col + 1
  4C | 75 17        |           BCS   DSP1         |  IF col > 79 THEN
  4E | 3E 00        |           MOV   COL,#$00     |
  50 | 2F 83 10     |           INC   ROW          |    col := 0
  53 | 3F 16        |           CMP   ROW,#MAXROW  |    row := row + 1
  55 | 75 0E        |           BCS   DSP1         |    IF row > 24 THEN
  57 | 3E 16        |           MOV   ROW,#MAXROW  |
  59 | CD __ __     |           CALL  SCRUP        |      row := 24
  5C | 3C 80 10     |           MOV   HL,SCRPNTR   |      ScreenUp
  5F | 1F 50 00     |           SUB   HL,#$0050    |      scrpntr := scrpntr - 80
  62 | BC 80 10     |           MOV   SCRPNTR,HL   |    ENDIF
  65 | EE 3C 80 10  | DSP1      MOV   HL,(SCRPNTR) |  ENDIF
  69 | F3           |           PUSH  HL           |
  6A | CD __ __     |           CALL  MOVCUR       |
076D | F0           | DSP2      RET                |  MoveCursor
                    |                              | END
                    |
                    | MAXCOL    EQU   $4F          |
                    | MAXROW    EQU   $16          |
                    |
1080                | SCRPNTR   DW    $C000        |
  82                | COL       DB    $00          |
  83                | ROW       DB    $00          |
```

รูปที่ 5 แสดงการเขียนโปรแกรมภาษาเครื่องขั้นที่สามคือ ให้แอดเดรสแก่คำสั่งทุกคำสั่งสำหรับแอดเดรสของตัวแปรนั้นความจริงต้องให้มาก่อน รูปนี้เพียงแต่นำมาพิมพ์รวมกันเท่านั้น

คงตัว MAXCOL กับ MAXROW ไว้ ถ้าเป็นโปรแกรมขนาดใหญ่จะเขียนไว้ในกระดาษอีกแผ่นหนึ่ง เพื่อสะดวกในตอนที่แปลงนิวมอนิกเป็นออปโค้ดด้วยมือ จะเห็นว่านำค่าไปใส่ตรง ๆ

ถัดลงไปเป็นการกำหนดตัวแปรทั้งสามตัว คือ SCRPNTR เป็นตัวแปรขนาด 2 ไบท์ และ COL กับ ROW ซึ่งเป็นตัวแปรขนาด 1 ไบท์ขอให้สังเกตว่าได้กำหนดค่าเริ่มต้นไว้ด้วย ซึ่งผมเคยกล่าวว่าจะให้โปรแกรมย่อย INIT เตรียมค่าให้ ซึ่งก็มีอีกวิธีหนึ่งคือให้ค่าโดยตรงซึ่งผมจะใช้วิธีหลังนี้ เพื่อแสดงถึงกรรมวิธีที่จะใช้วิธีนี้เมื่อใช้โปรแกรมแอสแซมเบลอร์ด้วย แอดเดรสที่กำหนดให้ก็สมมติว่าเป็นหน่วยความจำที่เรากำหนดไว้ให้เป็นที่อยู่ของตัวแปรทั้งหลายของโปรแกรมย่อยอื่นด้วย

การป้อนโปรแกรม

เมื่อเขียนโปรแกรมเสร็จแล้วก็ถึงขั้นตอนของการป้อนโปรแกรม ทำโดยนำออปโค้ดไปป้อนให้แก่หน่วยความจำ ณ แอดเดรสที่กำหนดไว้ เรียงตามลำดับจากแอดเดรสแรกไปจนจบโปรแกรม ขั้นตอนนี้ในสมัยก่อนหรือผู้ที่ใช้ซิงเกิลบอร์ดโดด ๆ ในปัจจุบันต้องใช้ความทรหดอดทนพอสมควร เพราะขั้นตอนที่จะผิดพลาดมีมาก เช่น เขียนโปรแกรมผิด แปลงเป็นออปโค้ดผิด ป้อนโปรแกรมผิด เมื่อผิดการตัดออกการแทรกทำได้ด้วยความยากลำบาก

สมัยนี้และโดยเฉพาะผู้ที่จะเขียนโปรแกรมภาษาเครื่องสำหรับภาษาสูงไม่ต้องวิตกครับเพราะเราจะใช้แอสเซมเบลอร์ แต่ก่อนที่จะไปถึงแอสเซมเบลอร์ขอให้สังเกตว่าพวกตัวแปรและค่าเริ่มต้นเราป้อนตรงด้วยมือ และก็ในโปรแกรมที่เป็นออปโค้ดแล้ว ไม่มีชื่อที่เราตั้งขึ้นใดเลยคงเหลืออยู่ และเป็นทั้งเมื่อใช้แอสเซมเบลอร์และคอมพายเลอร์ชื่อต่าง ๆ เหล่านี้เรากำหนดขึ้นมาเพื่อความสะดวกในการเขียนอ่านโปรแกรมที่เป็นนิวมอนิก

ใช้แอสเซมเบลอร์

แอสเซมเบลอร์ (Assembler) คือโปรแกรมตัวหนึ่งที่จะแปลงนิวมอนิกเป็นออปโค้ดพร้อมทั้งให้แอดเดรสด้วย บางท่านอาจสงสัยว่าภาษาเครื่องกับภาษาแอสเซมบลี (Assembly) ต่างกันตรงไหน ผมเองมองไม่เห็นความแตกต่าง ข้อที่จะเป็นข้อแตกต่างคือในการเขียนโปรแกรมภาษาแอสเซมบลี เราเขียนเฉพาะช่องเลเบิลกับนิวมอนิกรวมทั้ง

ช่องหมายเหตุ (หากมี) และต้องเป็นไปตามข้อกำหนดของแอสเซมเบลอร์แต่ละตัวซึ่งจะนำไปกล่าวในตอนต่อไป

ตัวแปรกับการผ่านค่า

ผู้ที่เขียนโปรแกรมในภาษาสูงโดยเฉพาะภาษาปาสคาลกับซี จะคุ้นเคยกับการกำหนดค่าคงตัว (constant) กับตัวแปร (variable) มาแล้ว ในภาษาเครื่องก็มีเช่นกันดังที่กล่าวมาแล้วคือ MAXCOL กับ MAXROW เป็นค่าคงตัว SCRPNTR เป็นตัวแปรขนาด 2 ไบท์ และเทียบได้กับตัวแปรพอยน์เตอร์ (pointer) ในภาษาซี ส่วน COL กับ ROW เป็นตัวแปรขนาด 1 ไบท์เทียบได้กับตัวแปรแบบไบท์ (Byte) ในเทอร์โบปาสคาล

ในภาษาสูงมีการผ่านค่าของตัวแปรจากโปรแกรมที่เรียกไปยังโปรแกรมที่ถูกเรียก ในภาษาเครื่องก็มีเช่นกันและมีวิธีดำเนินการอยู่ 4 แบบหลัก ๆ คือ

ผ่านทางรีจิสเตอร์ เป็นวิธีที่นิยมใช้ในโปรแกรมสั้น ๆ เขียนคนเดียว วิธีนี้ถึงแม้จะประหยัดเวลารันของเครื่อง แต่ตรวจแก้โปรแกรมได้ยากมาก หากใช้วิธีนี้ในตอนเรียกใช้ MOVCUR จะเขียนโปรแกรมดังนี้

```
DSP1  MOV HL, (SCRPNTR)
      CALL MOVCUR
DSP2  RET
```

สมมติว่าตำแหน่งเคอร์เซอร์ปัจจุบันเป็น $C010 ซึ่ง HL จะได้ค่านี้ เมื่อเขียนโปรแกรม MOVCUR จะใช้ค่านี้จากรีจิสเตอร์ HL

ผ่านทางตัวแปร วิธีการนี้เทียบได้กับการใช้ตัวแปรโกลบอล (global) ในภาษาปาสคาล เป็นแบบที่ตรวจแก้โปรแกรมยากเช่นกัน จะเขียนโปรแกรมได้ดังนี้

```
DSP1  CALL MOVCUR
DSP2  RET
```

ซึ่งผู้เขียนโปรแกรม MOVCUR ทราบดีอยู่แล้วว่าตำแหน่งของเคอร์เซอร์ในปัจจุบันอยู่ในตัวแปร SCRPNTR หรือคือที่แอดเดรส $1050 กับ $1051 จะอ่านมาดำเนินการเอง

ผ่านค่าทางสแต็ค วิธีนี้ใช้ในภาษาสูงทั่ว ๆ ไป เช่น ปาสคาลและซี เรียกว่า – pass by value คือ ผ่านค่าที่ต้องการจริง ๆ ไปทางสแต็ค คือที่ใช้ในโปรแกรม DSPCHAR ซึ่งขอลอกมาลงเพื่อเปรียบเทียบกันคือ

```
DSP1  MOV   HL, (SCRPNTR)
      PUSH  HL
      CALL  MOVCUR
DSP2  RET
```

ซึ่งใน MOVCUR จะดำเนินการดังนี้

```
MOVCUR POP   BC
       POP   HL
       PUSH  BC
```

ทำให้ MOVCUR ได้ตำแหน่งของเคอร์เซอร์ปัจจุบันคือ $C010 ตามที่ยกเป็นตัวอย่างใน HL ค่าที่ POP BC และ PUSH กลับไปคือ return address คือแอดเดรสตรง DSP2 ทั้งนี้เพราะในคำสั่ง CALL ซีพียูจะเก็บค่านี้ลงสแต็คโดยอัตโนมัติ

ผ่านแอดเดรสทางสแต็ค เป็นวิธีการที่ใช้ในภาษาสูงเช่นกัน เรียกว่า pass by reference วิธีการนี้จะผ่านแอดเดรสของหน่วยความจำที่เก็บค่าที่ต้องการหรือก็คือแอดเดรสของตัวแปรไปทางสแต็ค ใช้เมื่อจะมีการเปลี่ยนแปลงค่านี้ในโปรแกรมที่ถูกเรียก จึงไม่ได้นำมาใช้ใน DSPCHAR แต่ถ้าจะใช้ก็ใช้ได้เช่นกัน โดยเขียนโปรแกรมได้เป็น

```
DSP1  MOV   HL, SCRPNTR
      PUSH  HL
      CALL  MOVCUR
DSP2  RET
```

ซึ่งใน MOVCUR จะดำเนินการดังนี้

```
MOVCUR  POP   BC
        POP   HL
        PUSH  BC
        MOV   C, (HL)
        INC   HL
        MOV   B, (HL)
```

จากโปรแกรมท่อน DSP1 ค่าที่ให้แก่ HL คือค่าแอดเดรสของ SCRPNTR คือ $1050 ไม่ใช่ค่าตำบลที่ปัจจุบันของเคอร์เซอร์ ดังนั้นใน MOVCUR จึงต้องอ่านค่านี้มาให้แก่ BC จะทำให้ BC มีค่าเป็น $C010 จะเห็นว่าถ้าใน MOVCUR ต้องการจะเปลี่ยนค่าในตัวแปร SCRPNTR จะทำได้เช่นใช้คำสั่งว่า INC (HL) จะทำให้ค่าในตัวแปร SCRPNTR มีค่าเป็น $C011 แต่เมื่อผ่านค่าในแบบ pass by value ผู้เขียนโปรแกรมย่อย MOVCUR จะทำไม่ได้

ก่อนจบตอน

ที่ผมไม่ได้เขียนโปรแกรม MOVCUR ไปจนจบเพราะจะต้องไปเกี่ยวข้องกับฮาร์ดแวร์และยังไม่จำเป็น บทความของผมตอนนี้เพียงแต่ต้องการแสดงให้เห็นว่า ผู้ที่เขียนโปรแกรมภาษาเครื่องมีวิธีการเขียนโปรแกรมอย่างไร นับตั้งแต่ร่างจนถึงป้อนโปรแกรม วิธีการในการกำหนดค่าคงตัวและตัวแปรและการผ่านค่าซึ่งการผ่านค่าทางสแต็คเป็นวิธีที่ใช้กันในภาษาสูงและเราจะต้องนำไปใช้ด้วย

โปรดสังเกตด้วยว่าในการเขียนโปรแกรมได้แยกหน่วยความจำที่เก็บโปรแกรมกับที่เก็บข้อมูลออกจากกัน ที่เก็บโปรแกรมอยู่ที่แอดเดรส $0000 ถึง $0FFF ที่เก็บข้อมูลที่แอดเดรส $1000 ขึ้นไป เพื่อนำไปประกอบความเข้าใจเมื่อกล่าวถึง 8088 ซึ่งจะมีการแยกเป็น Code Segment คือหน่วยความจำที่เก็บโปรแกรม และ Data segment คือหน่วยความจำที่เก็บข้อมูล แต่ทั้งนี้ไม่ได้หมายความว่าจะนำมารวมกันไม่ได้

อีกเรื่องหนึ่งการเก็บข้อมูลในหน่วยความจำเรากล่าวว่าเก็บในนามของตัวแปร แต่ในทางปฏิบัติจะเป็นการกำหนดหน่วยความจำขึ้นมาหน่วยหนึ่งสำหรับตัวแปรตัวหนึ่งในขนาดกี่ไบท์ขึ้นกับข้อมูล และกำหนดชื่อให้มันเป็นตัวแปร ซึ่งชื่อของตัวแปรนี้จะหมายถึงแอดเดรสต้นของหน่วยความจำนั้น ทั้งในภาษาเครื่องและภาษาสูง ซึ่งการกำหนดหน่วยความจำนี้ถ้าเป็นภาษาสูงโปรแกรมตัวแปรภาษาเช่นคอมพายเลอร์จะเป็นผู้ดำเนินการ ถ้าเป็นภาษาเครื่องเราเป็นผู้ดำเนินการเองดังที่แสดงเป็นตัวอย่างในตอนนี้ ในตอนต่อไปจะแสดงถึงการเขียนภาษาเครื่องสำหรับซีพียูเบอร์ 8088 บนเครื่องไอบีเอ็มพีซีครับ